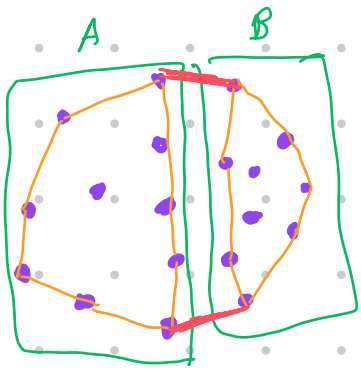
10 January 2025

DIVIDE AND CONQUER ALGORITHM

Recursive algorithm (it calls itself)



PSEUDO CODE:

Sort the points by x-coordinate.

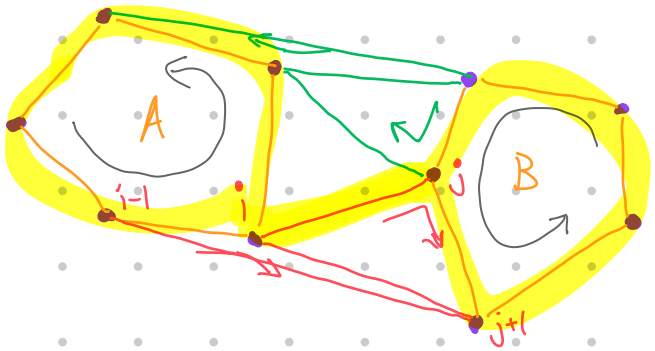
Let A be left half of the points and B be right half of the points.

Find convex hull of A.
Find convex hull of B
} Use the divide and conquer algorithm. (unless 3 points or fewer, in which case the convex hull is immediate)

Merge the convex hulls together.

Observation: If you traverse a convex hull counterclockwise, you make only left turns.

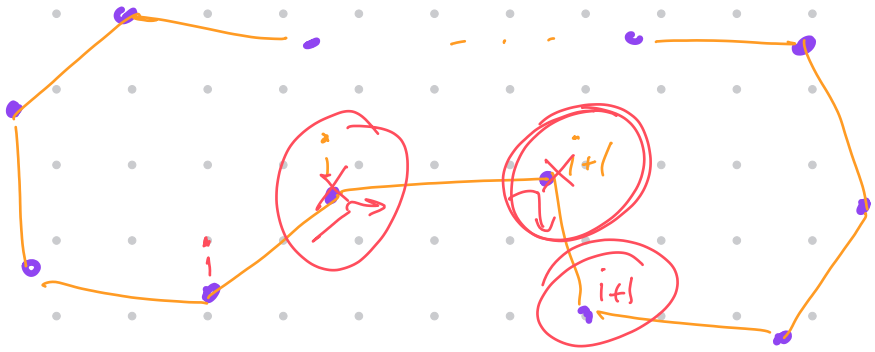
MERGE SUBROUTINE



linear
Complexity: $O(n)$

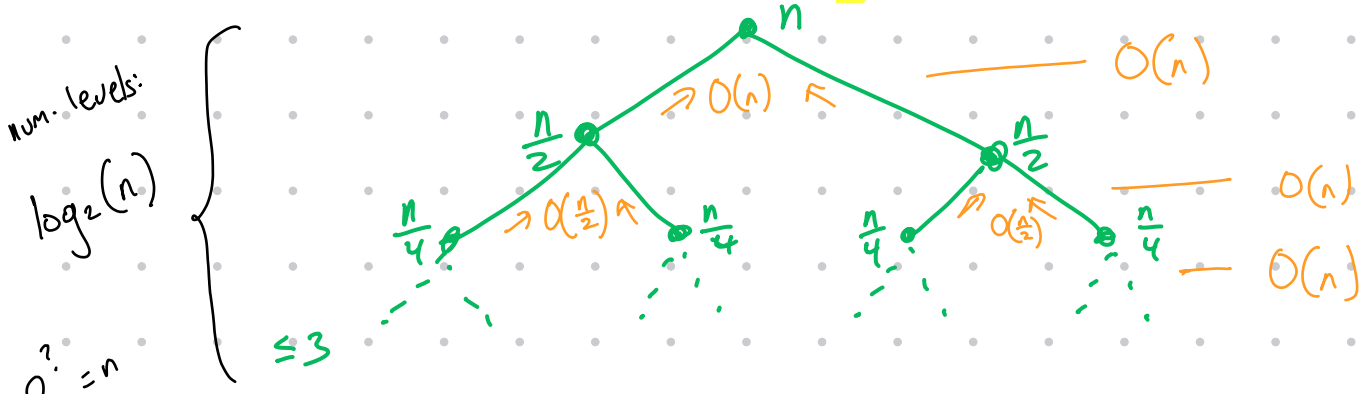
Insert an edge between rightmost point of A and the leftmost point of B.

Eliminate right turns to find the bottom and top edges connecting A and B.



Computational Complexity of Divide-and-Conquer

Sort the points: $O(n \cdot \log(n))$



Tree has $\log_2(n)$ levels, each of which requires $O(n)$ work.

Total computational complexity: $O(n \cdot \log n)$

Note: $\log_2(n) = \frac{\log_e(n)}{\log_e 2}$ const.

Divide-and-Conquer Algorithm

MATH 261 Computational Geometry

Input: a set S of n points in the plane, specified by xy-coordinates

Output: a list L of vertices of $\text{conv}(S)$ in counterclockwise order

Algorithm:

1. Sort the points by x-coordinate.

2. Let A be the left half of the points and B the right half of the points.

3. Find the convex hulls of A and B separately. If either set contains three points or fewer, this is trivial; otherwise use the divide-and-conquer algorithm.

4. Merge the convex hulls of A and B to form a single convex hull.

Answer the following questions:

1. Draw your own set of points. Work through the divide-and-conquer algorithm by hand to find the convex hull of your points.

2. How do we merge two convex hulls? Suppose you have left and right hulls, each specified by a list of vertices in counterclockwise order. Create an algorithm to compute the merged hull. How efficiently can you do this?

3. What is the overall computational complexity of the divide-and-conquer algorithm?

4. Download the incomplete Mathematical implementation of the divide-and-conquer algorithm from the course website. Complete the missing code to finish the implementation.

What is the optimal complexity for computing a convex hull in 2D?

THEOREM: Sorting a (arbitrary) list of n elements requires at least $c \cdot n \cdot \log(n)$ operations

why? There are $n!$ different arrangements of n items. Distinguishing among these requires $n \cdot \log(n)$ operations.

THEOREM: Finding the convex hull of n points in 2D requires $n \cdot \log(n)$ operations.

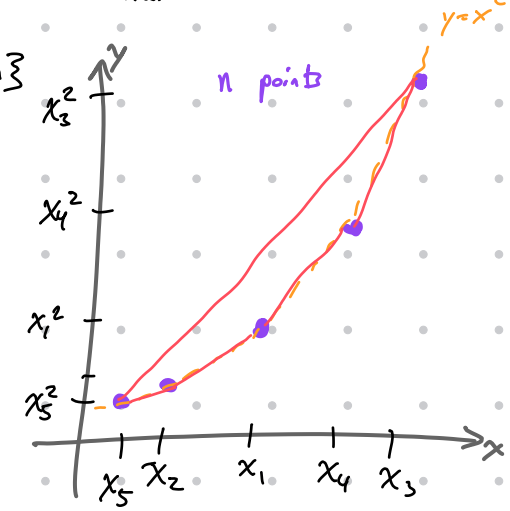
proof: If we could find the convex hull in fewer than $n \cdot \log(n)$ operations, then we could sort a list faster than $n \cdot \log(n)$.

Let $(x_1, x_2, \ldots, x_n)$ be an unsorted list of numbers.

Construct points (x_i, x_i^2) for $i \in \{1, 2, \ldots, n\}$

These points all lie on the parabola $y = x^2$, so they all lie on their convex hull.

Finding the hull returns the points as an ordered list (ccw order), solves the list sorting problem.

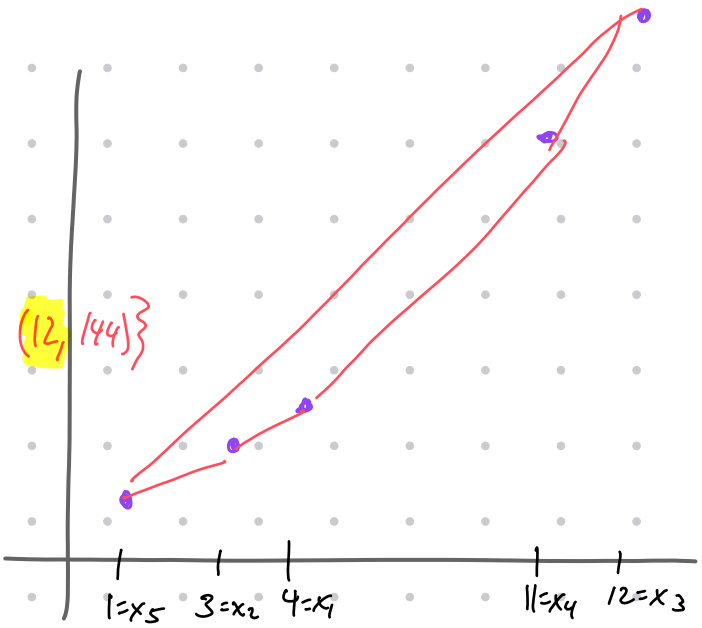


EXAMPLE:

4, 3, 12, 11, 1

hull:

$\{(1,1), (3,9), (4,16), (11,121), (12,144)\}$



Convex Hulls in 3D

MATH 261 Computational Geometry

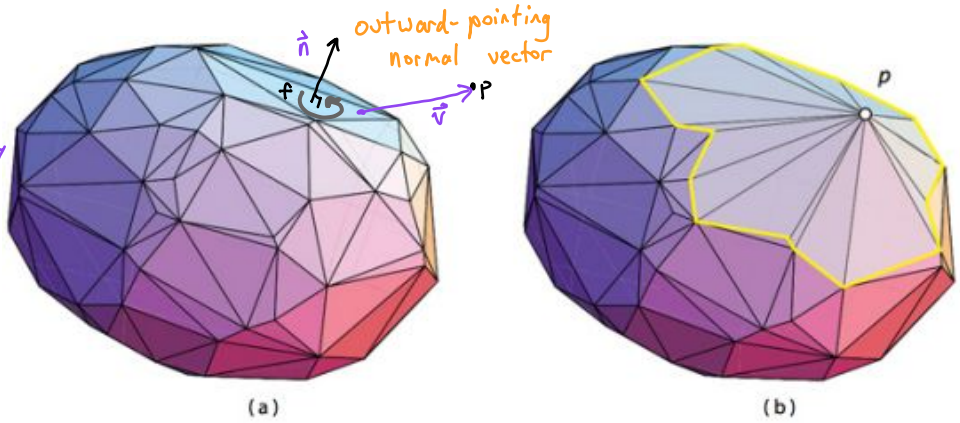
1. We can specify a 2D convex hull by a list of points, in order, around the convex hull. How can we specify a 3D convex hull in the memory of a computer? What data is required?

input: a list of points, each specified by (x, y, z) coordinates

store:
- for each point, its neighbors along hull edges
- list of (triangular) faces, each specifed by indexes of vertices
  - might include pointers to adjacent faces

2. How does the incremental algorithm extend to 3D? Specifically, if you have a 3D convex hull H and a point p outside of H, what steps are required to compute the convex hull of $H \cup p$?

dot product $\vec{n} \cdot \vec{v}$ is positive if and only if f is visible from p.



3. How does the divide-and-conquer algorithm extend to 3D? Specifically, if you have two disjoint 3D convex hulls A and B, what steps are required to compute the convex hull of $A \cup B$?

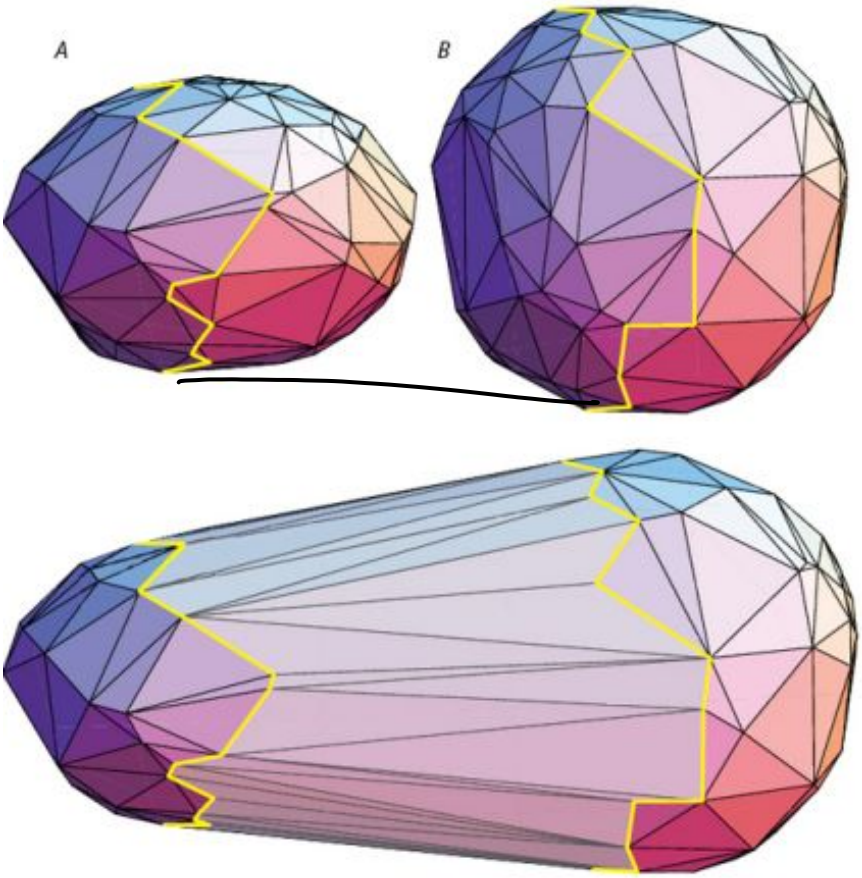


image credit: Devadoss and O'Rourke, *Discrete and Computational Geometry*

Convex hulls in 3D: Computational Complexity

- Incremental: $O(n^2)$
- Gift Wrapping: $O(n \cdot f) = O(n^2)$ where f is the number of faces: $f \leq 2n$
- Divide and Conquer: $O(n \cdot \log n)$
- Graham Scan: no one knows if this is possible